

Wandering in the Labyrinth of Thinking

– a minimalist cognitive architecture combining
reinforcement learning and deep learning

甄景贤 (King-Yin Yan), Ben Goertzel, and Juan Carlos Kuri Pinto

General.Intelligence@Gmail.com

Abstract. This is the first of a series of papers, introducing a minimalist cognitive architecture based on reinforcement learning and deep learning. The system consists of an iterative function whose role is analogous to the consequence operator (\vdash) in logic. Mathematically it is a Hamiltonian system, whose Lagrangian corresponds to the value of “desires” or “rewards” in the intelligent system. Techniques of classical logic-based AI can be transferred to the neural setting, the topic of the 2nd paper.

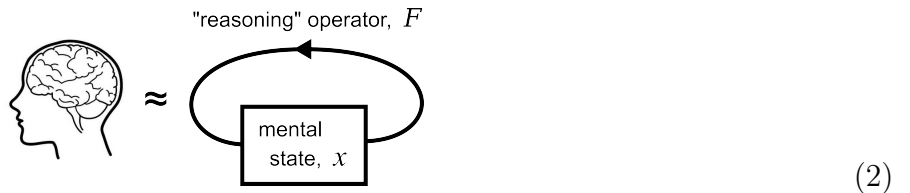
We introduce the basic architecture in §0, only this section and the basics of reinforcement learning (eg. see my tutorial [2]) are required to understand the rest of our theory. The “Hamiltonian stuff” is knowledge from existing theory and not our original contribution.

0 Main idea

The **metaphor** here is that of reinforcement learning controlling an autonomous agent to navigate the maze of “thoughts space”, seeking the optimal path:



The main idea is to regard “thinking” as a **dynamical system** operating on **mental states**:



For example, a mental state could be the following set of propositions:

- I am in my room, writing a paper for AGI-17.
- I am in the midst of writing the sentence, “I am in my room, ...”
- I am about to write a gerund phrase “writing a paper...”

Thinking is the process of **transitioning** from one mental state to another. Even as I am speaking now, I use my mental state to keep track of where I am at within the sentence's syntax, so that I can structure sentences grammatically.

The following 3 theories are actually synonymous:

- in artificial intelligence, **reinforcement learning (RL)**
- in operations research, **dynamic programming**
- in modern control theory, the **state space** description

1 Control theory / dynamical systems theory

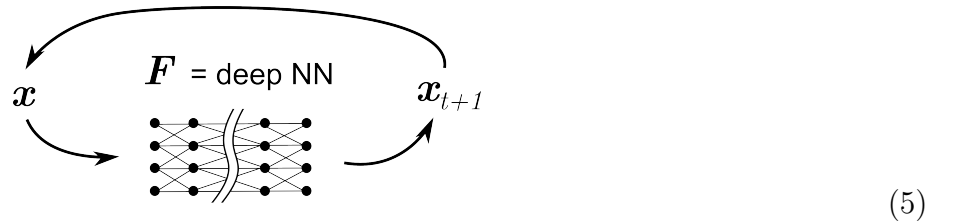
The cognitive state is a vector $\mathbf{x} \in \mathbb{X}$ where \mathbb{X} is the space of all possible cognitive states, the reasoning operator \mathbf{F} is an **endomorphism** (an **iterative map**) $\mathbb{X} \rightarrow \mathbb{X}$.

Mathematically this is a **dynamical system** that can be defined by:

$$\boxed{\text{discrete time}} \quad \mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t) \quad (3)$$

$$\boxed{\text{continuous time}} \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \quad (4)$$

In our cognitive architecture design, \mathbf{F} is implemented as a deep neural network (the word “deep” simply means “many layers”):



A **neural network** is a non-linear operator with many parameters (called “weights”):

$$F(\mathbf{x}) = \bigcirc(W_1 \bigcirc(W_2 \dots \bigcirc(W_L \mathbf{x})))$$

(6)

\bigcirc is a sigmoid-shaped non-linear function, applied component-wise to the vectors.

If continuous-time, \mathbf{f} can also be implemented as neural network, but \mathbf{f} and \mathbf{F} are different in nature, they are related by: $\mathbf{x}(t+1) = \mathbf{F}(\mathbf{x}(t))$. For ease of discussion, sometimes I mix discrete-time and continuous-time notations.

A **control system** is a dynamical system added with the control vector $\mathbf{u}(t)$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad (7)$$

The goal of control theory is to find the optimal $\mathbf{u}^*(t)$ function, such that the system moves from the initial state \mathbf{x}_0 to the terminal state \mathbf{x}_\perp .

A typical control-theory problem is described by:

$$\boxed{\text{state equation}} \quad \dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] \quad (8)$$

$$\boxed{\text{boundary condition}} \quad \mathbf{x}(t_0) = \mathbf{x}_0, \mathbf{x}(t_\perp) = \mathbf{x}_\perp \quad (9)$$

$$\boxed{\text{objective function}} \quad J = \int_{t_0}^{t_\perp} L[\mathbf{x}(t), \mathbf{u}(t), t] dt \quad (10)$$


and we seek the optimal control $\mathbf{u}^*(t)$.

According to control theory, the condition for **optimal path** is given by the Hamilton-Jacobi equation:

$$\boxed{\text{Hamilton-Jacobi equation}} \quad 0 = \frac{\partial J^*}{\partial t} + \min_u H \quad (11)$$

After the next section I will explain the meaning of J , L and H .

1.1 Reinforcement learning / dynamic programming

Reinforcement learning is a branch of machine learning that is particularly suitable for controlling an **autonomous agent** who interacts with an **environment**. It uses **sensory perception** and **rewards** to continually modify its **behavior**. The exemplary image you should invoke in mind is that of a small insect that navigates a maze looking for food and avoiding predators: 

A reinforcement learning system consists of a 4-tuple:

$$\boxed{\text{reinforcement learning system}} = (\mathbf{x} \in \text{States}, \mathbf{u} \in \text{Actions}, R = \text{Rewards}, \pi = \text{Policy}) \quad (12)$$

For details readers may see my *Reinforcement learning tutorial* [2].

U is the total rewards of a sequence of actions:

$$\begin{array}{ccc} \text{total value of state 0} & & \text{reward at time } t \\ & \swarrow & \swarrow \\ U(\mathbf{x}_0) & = & \sum_t R(\mathbf{x}_t, \mathbf{u}_t) \end{array} \quad (13)$$

For example, the value of playing a chess move is not just the immediate reward of that move, but includes the consequences of playing that move (eg, greedily taking a pawn now may lead to checkmate 10 moves later). Or, faced with delicious food, some people may choose not to eat, for fear of getting fat.

The central idea of **Dynamic programming** is the **Bellman optimality condition**. Richard Bellman in 1953 proposed this formula, while he was working at RAND corporation, dealing with operations research problems.

The **Bellman condition** says: “if we cut off a tiny bit from the endpoint of the optimal path, the remaining path is still an optimal path between the new endpoints.”

The **Bellman equation** governs reinforcement learning just as in control theory:

$$\boxed{\text{optimal path}} = \text{choose max reward on current path segment} + \boxed{\text{the rest of optimal path}} \quad (14)$$

In math notation:

$$U_t^* = \max_u \{ \boxed{\text{reward}(u, t)} + U_{t-1}^* \} \quad (15)$$

where U is the “long-term value” or **utility** of a path.

Conceptually, U is the **integration** of instantaneous rewards over time:

$$\boxed{\text{utility, or value } U} = \int \boxed{\text{reward } R} dt \quad (16)$$

- Intelligence is decomposed into **thinking** and **learning**.
- **Thinking** is governed by control theory (finding the best trajectory in “thoughts space”) under the constraints of correct reasoning, ie, knowledge.
- The iterative “thinking operator” is implemented as a deep-**learning** neural network (DNN). This DNN *constrains* the dynamics of thinking, and it represents the totality of *knowledge* in the system.

1.2 Connection with Hamiltonian mechanics

An interesting insight from control theory is that our system is a Hamiltonian dynamical system in a broad sense.

Hamilton’s **principle of least action** says that the trajectories of dynamical systems occurring in nature always choose to have their action S taking **stationary values** when compared to neighboring paths. The action is the time integral of the Lagrangian L :

$$\boxed{\text{Action } S} = \int \boxed{\text{Lagrangian } L} dt \quad (17)$$

From this we see that the Lagrangian corresponds to the instantaneous “rewards” of our system. It is perhaps not a coincidence that the Lagrangian has units of **energy**, in accordance with the folk psychology notion of “positive energy” when we talk about desirable things.

The **Hamiltonian** H arises when we consider a typical control theory problem; The system is defined via:

$$\text{state equation: } \dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] \quad (18)$$

$$\text{boundary condition: } \mathbf{x}(t_0) = \mathbf{x}_0, \mathbf{x}(t_{\perp}) = \mathbf{x}_{\perp} \quad (19)$$

$$\text{objective function: } J = \int_{t_0}^{t_{\perp}} L[\mathbf{x}(t), \mathbf{u}(t), t] dt \quad (20)$$

The goal is to find the optimal control $\mathbf{u}^*(t)$.

Now apply the technique of **Lagrange multipliers** for finding the maximum of a function, this leads to the new objective function:

$$U = \int_{t_0}^{t_{\perp}} \{L + \boldsymbol{\lambda}^T(t) [f(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}]\} dt \quad (21)$$

So we can introduce a new scalar function H , ie the Hamiltonian:

$$H(\mathbf{x}, \mathbf{u}, t) = L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T(t) f(\mathbf{x}, \mathbf{u}, t) \quad (22)$$

Physically, the unit of \mathbf{f} is velocity, while the unit of L is energy, therefore $\boldsymbol{\lambda}$ should have the unit of **momentum**. This is the reason why the phase space is made up of the diad of (position, momentum).

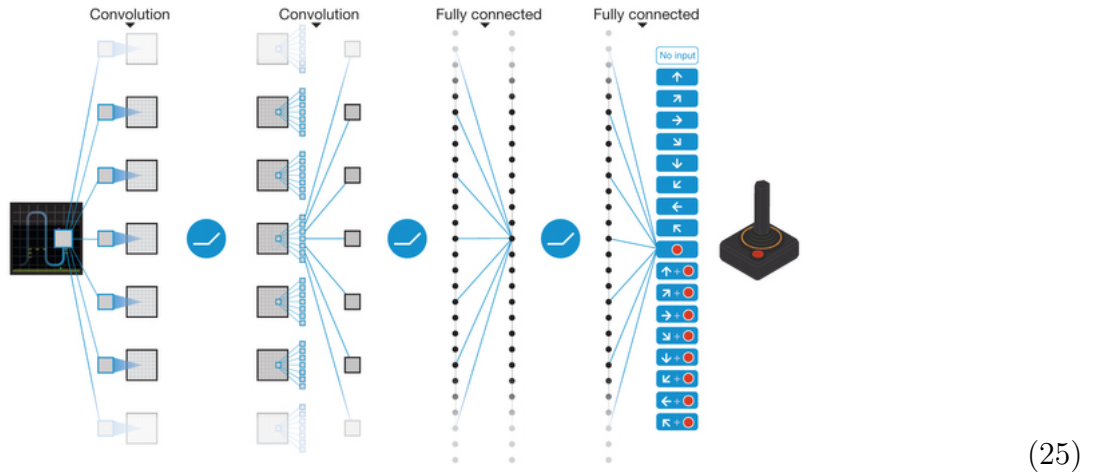
In its most general form we have the **Hamilton-Jacobi-Bellman equation**:¹

$$\boxed{\text{Hamilton-Jacobi-Bellman}} \quad 0 = \frac{\partial U^*}{\partial t} + \min_u H \quad (24)$$

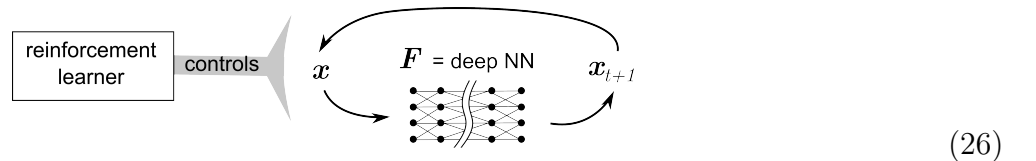
All these “physical” ideas flow automatically from our definition of **rewards**, without the need to introduce them artificially. But these ideas seem not immediately useful to our project, unless we are to explore **continuous-time** models.

1.3 Deep learning

The combination of deep learning with reinforcement learning, ie deep reinforcement learning (DRL), is very powerful. For example, DRL is able to play Atari games to human levels [1]. Their architecture is depicted in this diagram:



Recall our main architecture:

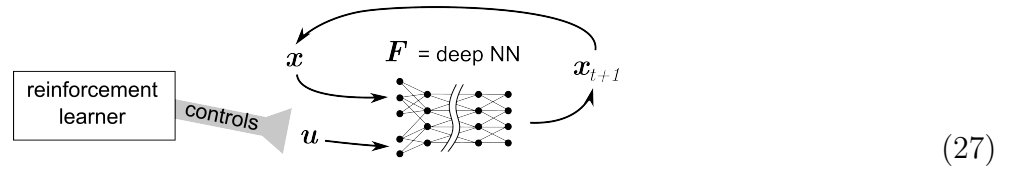


¹ To digress a bit, this equation is also analogous to the **Schrödinger equation** in quantum mechanics:

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[V(x, t) + \frac{-\hbar^2}{2\mu} \nabla^2 \right] \Psi(x, t). \quad (23)$$

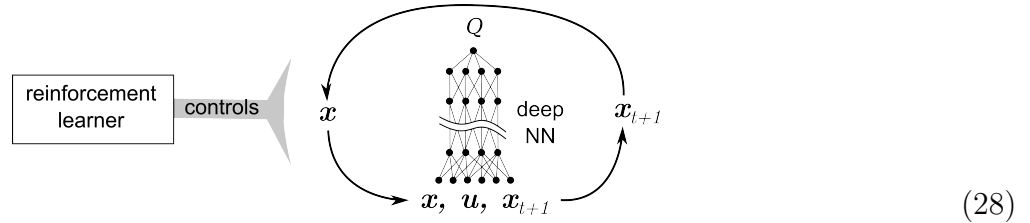
where Ψ is analogous to our U (perhaps Ψ is something that nature wants to optimize?)

The “transition function” neural network can also take an **action** parameter u :



But such a neural network requires a novel learning algorithm that is **reward-driven** rather than the traditionally **error-driven** back-propagation. Such an algorithm is difficult to design because it cannot rely on old-fashioned gradient descent.

One solution that avoids the difficulty is to make the neural network compute the Q-value:



This means that we can compute Q for any transition $x \xrightarrow{u} x'$. During the “action” (ie, “thinking”) stage, we hold x fixed, and search for (u, x') that maximizes Q ; This can be done by **stochastic gradient descent**. During the “learning” stage, we are given certain transitions (x, u, x') and we train the neural network to adjust Q via standard **Bellman update**.

Acknowledgement

In a forum discussion with Ben Goertzel dated 25 June 2014 on the AGI mailing-list: (agi@listbox.com), YKY asked: Why bother with neural networks, which typically require many neurons to encode data, when logic-based AI can represent a proposition with just a few symbols? Ben’s insight is that neural networks are capable of learning their own representations, and their learning algorithms are relatively speaking much faster. We have been working on “neo-classical” logic-based AI for a long time, and began to realize that inductive learning in logic (based on combinatorial search in a symbolic space) is perhaps *the bottleneck* in the entire logic-based paradigm. So we try to look for alternatives that might learn faster, though we would still emphasize that logic-based AI remains a viable approach to AGI.

References

1. Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra, and Riedmiller. Playing atari with deep reinforcement learning. *arXiv:1312.5602 [cs.LG]*, 2013.
2. King Yin Yan. Reinforcement learning tutorial
https://drive.google.com/file/d/0Bx3_S9SExak-SG4tUy1fSkVMSDA/view.